A Low-State Packet Marking Framework for Approximate Fair Bandwidth Allocation

Abhimanyu Das, Debojyoti Dutta, and Ahmed Helmy

***Abstract*—Misbehaving, non-congestion-reactive traffic is on the rise in the Internet. One way to control misbehaving traffic is to enforce local fairness among flows. Locally fair policies, such as Fair-Queueing and other fair AQM schemes, are inadequate to simultaneously control misbehaving traffic and provide high network utilization. We thus need to enforce globally fair bandwidth allocations. However, such schemes have typically been stateful and complex to implement and deploy. In this letter, we present a low state, lightweight scheme based on stateless fair packet marking at network edges followed by RIO queueing at core nodes, to control misbehaving flows with more efficient utilization of network bandwidth. Additionally, with low-state feedback from bottleneck routers, we show that, in practice, we can approximate global max-min fairness within an island of routers. We show, using simulations, that we can indeed control misbehaving flows and provide more globally fair bandwidth allocation.**

***Index Terms*—Fairness, protocol design, traffic marking.**

I. Introduction

In the current Internet, the amount of non-congestion-reactive (misbehaving) traffic is on the rise, which might lead to the poor performance of well behaved, congestion-reactive TCP [1] flows during congestion. One way to control the effects of such non-responsive traffic is to enforce locally fair bandwidth allocation among flows at individual routers. However, locally fair policies, such as Fair-Queueing and other well-studied approximately fair AQM schemes, such as FRED [2], RED-PD [3], CHOKe [4], and CSFQ [5], are inadequate to simultaneously control misbehaving traffic, and ensure high network-wide link utilization. For example, unresponsive flows receiving their locally fair share of bandwidth at a link will waste bandwidth if they are later bottlenecked at downstream nodes. We, thus, need to enforce globally fair bandwidth allocations. However, such schemes are typically stateful in nature [6], and, hence, hard to implement and deploy.

In this letter, we present a lightweight architecture based on low-state, fair packet marking [7] at network edges followed by RIO [8] queueing at core nodes, to control misbehaving flows. By using limited feedback from bottleneck routers, we show that we can approximate global max–min fairness, in practice, within an island of routers, and can provide high network-wide link utilization. Using detailed packet level simulations, we demonstrate that we can indeed control misbehaving flows with our architecture. We are not aware of any previous low-complexity distributed solution to the above problem.

Manuscript received December 24, 2003. The associate editor coordinating the review of this letter and approving it for publication was Prof. S. Pierre.

The authors are with the University of Southern California, Los Angeles, CA 90089 USA (e-mail: abhimand@usc.edu; ddutta@usc.edu; helmy@usc.edu).

Digital Object Identifier 10.1109/LCOMM.2004.833829

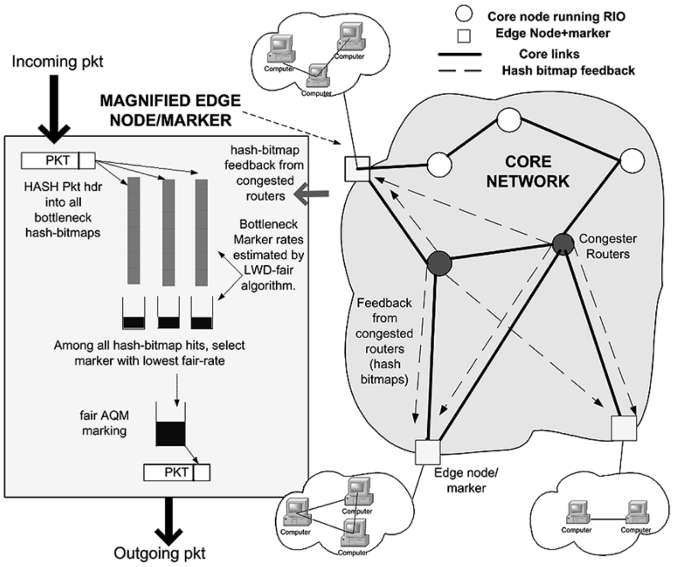
Core node running RIO
Edge Node+marker
Core links
Hash bitmap feedback
Incoming pkt
MAGNIFIED EDGE NODE/MARKER
PKT
HASH Pkt hdr into all bottleneck hash-bitmaps
hash-bitmap feedback from congested routers
Bottleneck Marker rates estimated by LWD-fair algorithm.
Among all hash-bitmap hits, select marker with lowest fair-rate
fair AQM marking
PKT
Outgoing pkt
Computer
CORE NETWORK
Congester Routers
Feedback from congested routers (hash bitmaps)
Edge node/ marker

Fig. 1. Our architecture.

II. Our Architecture

The core components of our architecture (see Fig. 1) include a set of traffic markers at the network edges and simple RIO [8] queueing at the core. These traffic markers, as described in our previous work [7], are based on low state or stateless AQMs, and can classify packets into two classes, IN and OUT. The markers are configured by a novel distributed marking algorithm using low-state feedback from congested routers (using summary data structures such as bit vectors or sketches) to approximate network-wide (globally) max–min fair IN token allocations. During congestion, a congested router uses RIO [8] as the AQM policy and preferentially drops OUT packets.

To design efficient traffic markers which mark flows fairly using little or no state, we leverage queueing and dropping policies of existing stateless AQM algorithms. As shown in [7], we view a token bucket specification at a marker as a queue, the marking policy as a queue dropping policy and marking packets as IN/OUT as queueing/dropping packets respectively. This novel analogy makes it easy to adapt well-studied low-state AQM schemes like CHOKe to design our markers which provide approximate fair IN token allocation among incoming flows at the network edges. We term these markers as **AQM-based markers**.

To approximate network-wide (globally) fair token-allocation among all flows, we make use of limited feedback from congested core routers to configure the target token rates for the markers using a light weight distributed fair (LWD-fair) algorithm described as follows. When a router is close to being congested, it starts hashing the flow ID of all the packets it sees into a summary data structure, described later. On congestion, the congested router sends its summary data structure to all the ingress nodes in the network. At each edge, the LWD-fair algorithm obtains the summaries from all the congested routers, and calculates an approximate globally fair marking rate for each congested router. For this, we need to obtain the approximate number of flows common to that congested router and itself, from the summary data structures. Each packet is marked with a target rate that is the minimum of the fair rates of all the bottlenecks traversed by the packet. This is determined from the summary data structures as shown in Algorithm 1. See Fig. 1 for a summary of our architecture. Thus, each edge has one AQM based marker per bottleneck, and packets of a flow are marked by the marker corresponding to its tightest bottleneck.

We maintain a summary data structure at each router using a combination of multi-resolution bitmaps [9] and bloom-filters [10]. Our data structure requires low processing time per arriving packet (each packet is hashed on its flow-id to update a few bits in the bitmap [9]) and low space complexity. For example, if N is number of flows, and ϵ is the allowed error, the total number of bits $= O[(\log N\epsilon^2)/\epsilon^2]$ for a sophisticated multiresolution bitmap. Estan *et al.* show that using multi-resolution bitmaps with 8 kbits, average errors for counting up to 1 000 000 flows are only 3%. Use of bloom filters can at most increase the space by a constant factor. It is clearly feasible for most routers to store this amount of information and propagate it to all edge nodes upon congestion, using routing or multicast messages. These summary structures for each bottleneck link can help us to calculate both the number of flows at a link, and the number of flows intersecting at two links, using low state feedback.

It can be shown that our LWD-fair algorithm reduces the per-flow state required by the centralized global max-min algorithm, using our summary data structures. The basic feedback requirement from the core to the edge nodes, for our LWD-algorithm to calculate global fairness is threefold. We first need an estimate of the number of flows at any given link. Secondly we need an estimate of the number of flows common to two links. Third, we need a mechanism to find out which bottlenecks are traversed by a given packet.

III. Evaluation

In this section, we compare the performance of our architecture against that due Fair Queueuing and due to the theoretical centralized max-min fair algorithm, using detailed packet level simulations using the ns-2 [11] network simulator. The topology used for most of our experiments is depicted in Fig. 2. We use multiple bottlenecks and both UDP as well as a mixture of UDP and TCP flows for our evaluation. The source nodes are s_i and the destination or sink nodes are d_i. The two edge nodes are e_0 and e_1 connected to e_2. The source nodes s_0 to s_i and n_0 to n_k are connected to the edge nodes e_0, e_1, respectively, and inject traffic into the network core c. The core is connected to the edge e_3 which is further connected to the sink nodes d_i's. The source node s_3 is used to generate background traffic in form of many TCP flows carrying bulk traffic. Note that the bottleneck link is ce_2 with a capacity 1 Mb/s. All the other links have a capacity of

Algorithm 1: LWD-Fair Algorithm

```
Data    : $local$: edge's incoming link;
          $Markerrate[i]$: tokenbucket rate for bottleneck i's marker;
          $Marker[i]$: tokenbucket marker for bottleneck i;
          $DS[i]$: flow summary received from bottleneck i;
          $Card(DS[i])$: num of flows calculated from summary DS[i] for bottleneck i;
          $Bw[i]$: link bandwidth of bottleneck i;
          $M$: set of bottleneck links that sent feedback;
          $P$: temporary flow datastructure initially empty;
          $N[i]$: initialized to Cardinality(DS[i]);
          $Isect(DS[i], DS[j])$: datastructure containing the intersection of flows of bottlenecks i and j);
          $Union(DS[i], DS[j])$: datastructure containing the union of flows belonging to bottlenecks i and j);
          $Diff(DS[i], DS[j])$ datastructure containing the set difference of flows belonging to bottlenecks i and j;
          $Member(f, DS[i])$: boolean variable that is true if flow $f$ passes through bottleneck i;
if event == new summary-datastructure feedback then
    cnt=0;
    while M is not empty do
        foreach i such that i is in M do
            $est\_rate[i] = \frac{Bw[i]}{N[i]}$;
        tightlink = link s.t. $est\_rate[link]$ is min in M;
        $fairrate[tightlink]=est\_rate[tightlink]$;
        remove tightlink from M;
        P = Union(P, DS[tightlink]) //add flows of tightlink to P;
        foreach j in M do
            $temp =$;
            $Card(Diff(Isect(DS[tightlink], DS[j]), P))$;
            $Bw[j] = Bw[j] - temp * fairrate[tlink]$;
            $N[j] = N[j] - temp$;
        cnt++;
    $Markerrate[i] = Card(Isect(DS[local], DS[i])) * fairrate[i]$
if event == recvd new pkt(p) then
    MarkerSet = All links l in M such that Member(p,DS[l])=true;
    MarkerBottleneck = m ∈ Markerset such that fairrate[m] is minimum among all links in MarkerSet;
    Mark(p,Marker[m]);
```

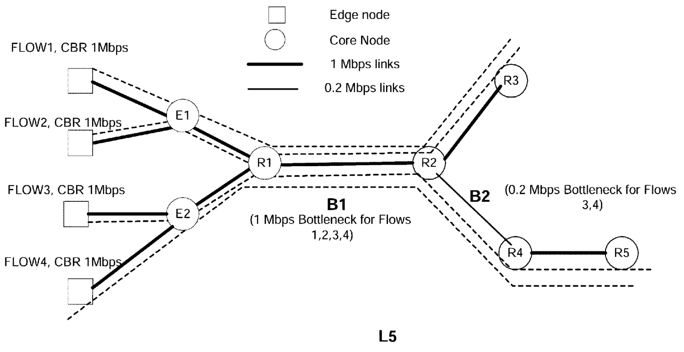

Fig. 2. Simulation topology.

CBR Flow Throughputs (Mbps)	***Flow 1***	***Flow 2***	***Flow 3***	***Flow 4***
Theoretical global max-min fair rates	0.4	0.4	0.1	0.1
LWD-fair algorithm with CAM marking	0.42	0.41	0.083	0.082
Fair Queueing at each node	0.249	0.249	0.099	0.099

Fig. 3. Performance of LWD-fair—UDP.

50 Mb/s. Each link has a propagation delay of 5 ms and a buffer size of 50 packets. We use TCP Reno with a window size of 20 packets. The CAM markers are configured with a total CIR of the bottleneck link bandwidth.

We first consider a simple example with four UDP flows. All the flows are 1-Mb/s UDP sources, and there are two bottlenecks—a 1-Mb/s link and a 0.2-Mb/s link. We compare the end to end throughput obtained by the UDP flows in our framework, with that obtained by running fair queueing at all nodes. We observe in Fig. 3, that using our LWD-fair marking algorithm, the bandwidth obtained by the flows actually comes very close to that theoretically predicted by global max-min fairness algorithm (denoted by FS in our graphs) while local fair-queueing (denoted by FQ in our graphs) does not provide global fairness, and, in fact, provides lower total throughput than our scheme. Thus we see that for non-congestion reactive flows, fair-queueing does not provide high network utilization, while LWD-fair marking does. A corollary of this result is that since our scheme here approximates global fairness, all misbehaving flows in the network are, therefore, automatically restricted to their fair levels.

We now consider a more complex case using the same topology as earlier, but using a mixture of TCP and UDP flows in the network. We have a fixed set of four TCP flows aggregating at edge E1 and passing through node R3, while we vary the number of 1-Mb/s UDP flows (from 2 to 10), which aggregate at edge E2 and leave through R5. There are thus two bottlenecks, the 1-Mb/s bottleneck for all flows, and the subsequent 0.2 Mb/s bottleneck for the UDP flows later. As a measure of network utilization and fairness in this scenario, we compare the average TCP throughput obtained in our scheme, with that using Fair Queueing, and the theoretical globally fair algorithm. The globally fair values for the TCP flows should be $(1 - 0.2)/4 = 0.2$ Mb/s since the UDP flows are bottlenecked later at a 0.2-Mb/s link. The results obtained in our scheme are very similar to the theoretical global max-min values. However with FQ, we see that the TCP performance degrades progressively as the number of UDP flows increase, due to FQ's attempt to provide local fairness in spite of the fact that the UDP flows will not be able to use their allocated fair-share further ahead. We therefore illustrate that in the absence of global fairness, locally fair router schemes may not provide high network utilization if some flows are non-congestion-reactive. The graphs in Fig. 4 indicate that our LWD-fair algorithm based on AQM markers can help to solve this problem.

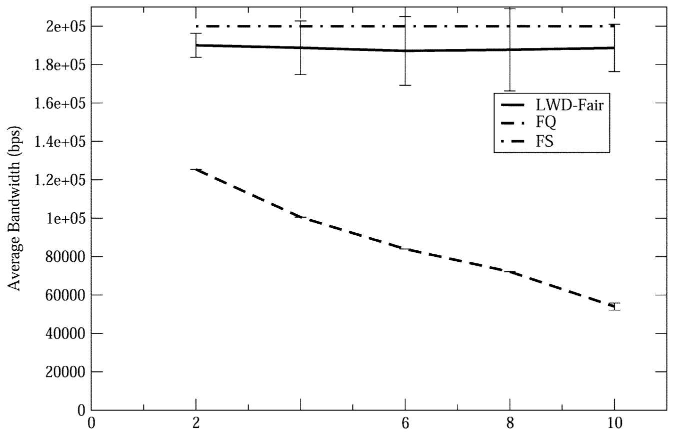

Fig. 4. Performance of LWD-fair—$\mathrm{TCP} + \mathrm{UDP}$.

IV. Conclusion

In this letter, we introduced a lightweight architecture that can provide efficient network-wide max-min fair bandwidth allocation along with high network utilization. Our framework consists of marking packets at edges using a LWD-fair marking algorithm that uses low-state feedback from core routers and AQM-based markers to allocate bandwidth to flows in an approximately global max-min fair manner. As a by-product, we note that this seemingly simple architecture can punish misbehaving flows and consequently improve the throughput of well-behaved TCP flows dramatically.

Our ongoing work includes investigating sophisticated summary data structures for a better approximation of the number of common active flows between two router, and more extensive validation of our architecture (i.e., for larger networks and larger number of bottlenecks/flows). We are also evaluating different mechanisms and appropriate frequencies for disseminating the feedback.

References

[1] J. Postel, "Transmission control protocol," in *Internet Request for Comments RFC*, vol. 793, Sept. 1981.

[2] D. Lin and R. Morris, "Dynamics of random early detection," in *Proc. SIGCOMM'97*, Sept. 1997, pp. 127–137.

[3] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling high-bandwidth flows at the congested router," in *Proc. Int. Conf. Network Protocols (ICNP'01)*, 2001.

[4] R. Pan, B. Prabhakar, and K. Psounis, "A stateless active queue management scheme for approximating fair bandwidth allocation," in *Proc. IEEE INFOCOM 2000*, 2000.

[5] I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks," in *Proc. SIGCOMM*, 1998.

[6] A. Charny, D. Clark, and R. Jain, "Congestion control with explicit rate indication," in *Proc. IEEE Int. Conf. on Communications*, 1995, pp. 1954–1963.

[7] A. Das, D. Dutta, and A. Helmy, "Fair stateless aggregate marking techniques using active queue management techniques," in *IEEE/IFIP MMNS*, Oct. 2002.

[8] D. D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," in *IEEE/ACM Trans. Networking*, vol. 6, Aug. 1998, pp. 362–373.

[9] C. Estan, G. Varghese, and M. Fisk, "Bitmap algorithms for counting active flows on high speed links," in *Proc. ACM IMC*, Oct. 2003.

[10] A. Broder and M. Mitzenmacher, Network applications of Bloom filters: A survey.

[11] UCB/LBNL/VINT. The NS2 Network Simulator. [Online]. Available: http://www.isi.edu/nsnam/ns/.